#13

PATENT APPLICATION
Docket No.: NC 84,781

RECEIVED
CENTRAL FAX CENTER
MAY 28 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the application of: Dennis
Serial No.: 09/715,772
Filed: 11/17/2000
For: MULTI-THREAD PERIPHERAL PROCESSING USING DEDICATED PERIPHERAL BUS
Examiner: King, Justin
Art Group Unit: 2181

OFFICIAL

Honorable Commissioner of Patents
PO Box 1450
Alexandria, VA 22313-1450

May 28, 2004

TRANSMITTAL LETTER

Sir:

In response to the Advisory Action of 04/16/2004, please enter the attached three copies of an Appeal Brief into the above-stated case for consideration by the Examiner. The appeal brief fee is paid by the attached Fee Transmittal

In the event that a fee is required, please charge the fee to Deposit Account No. 50-0281, and in the event that there is a credit due, please credit Deposit Account No. 50-0281.

Respectfully submitted,

05/28/2004 EWILLIAM 00000002 500281 09715772
01 FC:1402 330.00 DA

Joseph T. Grunkemeyer
Reg. No. 46,746
Phone No. 202-404-1556
Office of the Associate Counsel
(Patents), Code 1008.2
Naval Research Laboratory
4555 Overlook Ave, SW
Washington, DC 20375-5325

CERTIFICATION OF FACSIMILE TRANSMISSION

I certify that this paper is being facsimile transmitted to the Patent and Trademark Office on the date shown below.

5/28/04
Date

Joseph T. Grunkemeyer

OFFICIAL

Fax transmission from:
U.S. Naval Research Laboratory
Associate Counsel Intellectual Property, Code 1008.2
4555 Overlook Avenue, SW
Washington, DC 20375-5325

From: Joe Grunkemeyer
Phone: (202) 404-1556
Fax: (202) 404-7380

To: Examiner Justin King
Art Unit 2181
Fax: (703) 872-9306

Date: 05/28/2004

Cover Sheet + 56 Pages

Message:

RE: Patent Application No.: 09/715,772

Filed: 11/17/2000

Inventor: Dennis

Docket No.: NC 84,781

Fee Transmittal – 1 page

Transmittal Letter – 1 page

Appeal Brief – 3 copies, 18 pages each

PATENT APPLICATION
Docket No.: NC 84,781

#13
6/3/04

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the application of: Dennis
Serial No.: 09/715,772
Filed: 11/17/2000
For: MULTI-THREAD PERIPHERAL PROCESSING USING DEDICATED PERIPHERAL BUS
Examiner: King, Justin
Art Group Unit: 2181

Honorable Commissioner of Patents
PO Box 1450
Alexandria, VA 22313-1450

May 28, 2004

APPEAL BRIEF

Sir:

The present appeal is taken from the 12/29/2003 final rejection of claims 1-41 (all claims presently under consideration). A copy of the claims on appeal, as amended in the 02/06/2004 AMENDMENT AFTER FINAL REJECTION, is attached as the APPENDIX.

REAL PARTY IN INTEREST

The real party in interest is the United States Government, as represented by the Secretary of the Navy.

RELATED APPEALS AND INTERFERENCES

Appellant is unaware of any other appeals or interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-41 are presently pending. All claims have been rejected.

STATUS OF AMENDMENTS

One amendment was filed after final rejection on 02/06/2004. The Advisory Action of 02/24/2004 indicated that for purposes of appeal, the amendment would be entered.

SUMMARY OF THE INVENTION

The invention is an apparatus and method for performing peripheral operations in a multi-thread processor. The apparatus (claims 1-13) comprises a peripheral bus 260 (Figs. 2, 3, and 6) coupled to a peripheral unit 130 (Figs. 1, 2, and 6) (p. 4, lines 3-4) and a processing slice 310 (Figs. 3 and 6) coupled to the peripheral bus 260 (p. 4, lines 5-6). The peripheral bus 260 transfers peripheral information including a command message 500 (Fig. 5) specifying a peripheral operation (p. 4, lines 3-5). The processing slice 310 executes a plurality of threads (p. 4, line 5) comprising instructions (p. 8, lines 23-24). The threads include a first thread sending the command message 500 to the peripheral unit 130 (p. 4, lines 6-8). The processing slice 310 comprises a functional unit 450 (Fig. 4) to perform a register operation specified in the instructions in each thread (p. 12, lines 5-7). The processing slice executes the instructions from more than one of the plurality of threads concurrently in a clock cycle (p. 8, lines 23-25).

The method (claims 14-26) comprises transferring the peripheral information to the peripheral unit 130 via the peripheral bus 260 (p. 4, lines 3-5), and executing the plurality of threads by the processing slice 310 (p. 5, lines 1-2). The invention also includes processing systems (claims 27-41) incorporating the peripheral bus 260 and processing slice 310 (p. 6, line 7-p. 9, line 4).

ISSUES

A. Whether claims 1-12, 14-25, 27-38, 40, and 41 are unpatentable under 35 U.S.C. § 103(a) over the combination of Bucher (US Pat. No. 5,421,014) and Motomura (US Pat. No. 5,815,727).

B. Whether claims 13, 26, and 39 are unpatentable under 35 U.S.C. § 103(a) over the combination of Bucher, Motomura, and Hiraoka (US Pat. No. 5,418,917).

GROUPING OF CLAIMS

With regard to each ground of rejection, the rejected claims stand or fall together.

ARGUMENT

A. Claims 1-12, 14-25, 27-38, 40, and 41 define patentable subject matter over the

combination of Bucher and Motomura.

In order to make a *prima facie* case of obviousness, the references must disclose each limitation of the claims. *In re Royka*, 180 U.S.P.Q. 580, 490 F.2d 981 (CCPA 1974). The processing slice recited in each independent claim (1, 14, 27, 40, and 41) is not disclosed in either Bucher or Motomura. The processing slice of the present invention has a functional unit that can perform operations from each of the plurality of simultaneously executing threads. The processing slice is able to dispatch an instruction from any currently executing thread to any of the functional units within the processing slice. Thus, each functional unit within a given processing slice is shared among multiple threads that are simultaneously executing in that processing slice. This is an efficient design in that fewer functional units are needed. It is unlikely that all currently executing threads would need constant use of the functional units. Since each thread can use any functional unit, there can be full or near full utilization of functional units with little to no delay to wait for a functional unit to be available.

Bucher discloses a software architecture for implementing multi-thread control of a peripheral interface, specifically a SCSI interface. The software operates at the driver level and manages multiple peripheral requests. A higher level program sends a peripheral request to the driver. When the operation is complete, the driver sends the result to the high level program.

As Bucher addresses software, no internal details of the processor are disclosed. In a conventional processor, instructions from only one thread are being executed at any time. Although the term "multi-thread mode" is used (Abstract), there is no disclosure that the multiple threads are executing simultaneously or that multiple instructions are executed concurrently in one clock cycle. In fact, Bucher states that the high level code must wait for a return from the low-level driver to continuing issuing commands (col. 3, lines 52-63). Thus, there is no simultaneous execution of multiple threads. Further, the Examiner admitted in the Advisory Action of 02/24/2004 (continuation of 10) that the rejection used Motomura to support concurrent processing.

Motomura discloses a multi-thread parallel processor system having a plurality of processors and an ordered multithread executing system (Fig. 1). The ordered multithread executing system determines which thread will execute on each processor.

The Examiner stated that Fig. 1 of Motomura as a whole is equivalent to Applicant's

processing slice (Advisory Action of 02/24/2004, continuation of 10). Although the system as a whole is capable of simultaneous execution of multiple threads, there is no sharing of functional units among threads. Motomura discloses that each processor can execute only one thread at a time, in that the thread must go into a waiting or completed state before another thread is assigned to the processor (col. 8, lines 40-51). Further, there are no connections disclosed between processors. Each processor can communicate only with the ordered multithread execution system and the memory device (Fig. 1). The result of this configuration is that the functional units of any given processor are dedicated only to the one thread that can execute on that processor at a time. An idle functional unit cannot be allocated to a thread executing on another processor. In the present application, it is specifically recited in the claims that the functional unit can perform a register operation specified in the instructions in *each* of the threads, which are simultaneously executing. Motomura lacks this capability because any functional unit within a processor can perform operations from only *one* thread.

B. Claims 13, 26, and 39 define patentable subject mater over the combination of Bucher, Motomura, and Hiraoka.

Claims 13, 26, and 39 recite an instruction fetch unit, an instruction buffer, and an instruction decoder and dispatcher. These claims depend on the independent claims and include the processing slice.

Hiraoka discloses a method and apparatus for controlling a conditional branch instruction in a pipeline type data processing apparatus. Hiraoka does not disclose simultaneous execution of multiple threads, and so, does not disclose a processing slice. As none of the references discloses the processing slice, there is no *prima facie* case of obviousness.

CONCLUSION

For the reasons stated above, reversal of the rejections under 35 U.S.C. § 103 are earnestly solicited.

In the event that a fee is required, please charge the fee to Deposit Account No. 50-0281,

and in the event that there is a credit due, please credit Deposit Account No. 50-0281.

Respectfully submitted,

Joseph T. Grunkemeyer
Reg. No. 46,746
Phone No. 202-404-1556
Office of the Associate Counsel
(Patents), Code 1008.2
Naval Research Laboratory
4555 Overlook Ave, SW
Washington, DC 20375-5325

APPENDIX-THE CLAIMS ON APPEAL

1. An apparatus comprising:

a peripheral bus coupled to a peripheral unit to transfer peripheral information including a command message specifying a peripheral operation; and

a processing slice coupled to the peripheral bus to execute a plurality of threads comprising instructions, the plurality of threads including a first thread sending the command message to the peripheral unit;

wherein the processing slice comprises a functional unit to perform a register operation specified in the instructions in each of the plurality of threads; and

wherein the processing slice executes the instructions from more than one of the plurality of threads concurrently in a clock cycle.

2. The apparatus of claim 1 wherein the peripheral unit is one of an input device and an output device.

3. The apparatus of claim 1 wherein the peripheral operation is one of an input operation and an output operation.

4. The apparatus of claim 1 wherein the command message includes at least one of a message content, a peripheral address identifying the peripheral unit, and a command code specifying the peripheral operation.

5. The apparatus of claim 1 wherein the peripheral information includes a response message sent from the peripheral unit to the processing slice, the response message indicating the peripheral operation is completed.

6. The apparatus of claim 5 wherein the response message includes at least one of a thread identifier identifying the first thread, an operation result of the peripheral operation, a data register address specifying a data register in the processing slice to store the operation result, and a length indicator indicating length of the response message.

7. The apparatus of claim 6 wherein the peripheral bus comprises:

a bi-directional bus to transfer the command message from the processing slice to the peripheral unit and the response message from peripheral unit to the processing slice.

8. The apparatus of claim 1 wherein the processing slice disables the first thread after sending the command message if the command message is a wait instruction.

9. The apparatus of claim 1 wherein the first thread continues to execute after sending the command message if the command message is a non-wait instruction.

10. The apparatus of claim 8 wherein the processing slice enables the first thread after receiving the response message from the peripheral unit if the first thread was disabled.

11. The apparatus of claim 1 wherein the processing slice comprises:

an instruction processing unit to process instructions fetched from a program memory; and

a thread control unit coupled to the instruction processing unit to manage initiating and termination of at least one of the plurality of threads.

12. The apparatus of claim 11 wherein the processing slice further comprises:

a memory access unit coupled to the instruction processing unit to provide access to one of a plurality of data memories via a data memory switch, the memory access unit having a plurality of data base registers, each of the data base registers corresponding to each of the threads; and

a register file coupled to the instruction processing unit and a peripheral message unit having a plurality of data registers, each of the data registers corresponding to each of the threads.

13. The apparatus of claim 12 wherein the instruction processing unit comprises:

an instruction fetch unit to fetch the instructions from the program memory using a plurality of program counters, each program counter corresponding to each of the threads;

an instruction buffer coupled to the instruction fetch unit to hold the fetched instructions; and

an instruction decoder and dispatcher coupled to the instruction buffer to decode the instructions and dispatch the decoded instructions to one of the memory access unit, the functional unit, and the peripheral unit.

14. A method comprising:

transferring peripheral information to a peripheral unit via a peripheral bus, the peripheral information including a command message specifying a peripheral operation; and

executing a plurality of threads comprising instructions by a processing slice, the plurality of threads including a first thread sending the command message to the peripheral unit;

wherein the processing slice comprises a functional unit to perform a register operation specified in the instructions in each of the plurality of threads; and

wherein the processing slice executes the instructions from more than one of the plurality of threads concurrently in a clock cycle.

15. The method of claim 14 wherein the peripheral unit is one of an input device and an output device.

16. The method of claim 14 wherein the peripheral operation is one of an input operation and an output operation.

17. The method of claim 14 wherein the command message includes at least one of a message content, a peripheral address identifying the peripheral unit, and a command code specifying the peripheral operation.

18. The method of claim 14 wherein the peripheral information includes a response message sent from the peripheral unit to the processing slice, the response message indicating the peripheral operation is completed.

19. The method of claim 18 wherein the response message includes at least one of a thread identifier identifying the first thread, an operation result of the peripheral operation, a data register address specifying a data register in the processing slice to store the operation result, and a length indicator indicating length of the response message.

20. The method of claim 19 wherein transferring the peripheral information comprises:

transferring the command message from the processing slice to the peripheral unit and the response message from peripheral unit to the processing slice via a bi-directional bus.

21. The method of claim 14 wherein executing the plurality of threads comprises disabling the first thread after sending the command message if the command message is a wait instruction.

22. The method of claim 14 wherein executing the plurality of threads comprises continuing executing the first thread after sending the command message if the command message is a non-wait instruction.

23. The method of claim 21 wherein executing the plurality of threads comprises enabling the first thread after receiving the response message from the peripheral unit if the first thread was disabled.

24. The method of claim 14 wherein executing the plurality of threads comprises:

processing instructions fetched from a program memory by an instruction processing unit;

managing initiating and termination of at least one of the plurality of threads by a thread control unit.

25. The method of claim 24 wherein executing the plurality of threads further comprises:

accessing to one of a plurality of data memories by a memory access unit via a data memory switch, the memory access unit having a plurality of data base registers, each of the data base registers corresponding to each of the threads; and

storing data in a register file having a plurality of data registers, each of the data registers corresponding to each of the threads.

26. The method of claim 25 wherein processing instructions comprises:

fetching the instructions from the program memory using a plurality of program counters by an instruction fetch unit, each program counter corresponding to each of the threads;

holding the fetched instructions in an instruction buffer; and

decoding the instructions and dispatching the decoded instructions by an instruction decoder and dispatcher to one of the memory access unit, the functional unit, and the peripheral unit.

27. A processing system comprising:

a plurality of banks of data memory;

a data memory switch coupled to the banks of data memory;

a program memory to store a program;

a peripheral bus coupled to a peripheral unit to transfer peripheral information including a command message specifying a peripheral operation; and

a processing slice coupled to the peripheral bus to execute a plurality of threads comprising instructions, the plurality of threads including a first thread sending the command message to the peripheral unit;

wherein the processing slice comprises a functional unit to perform a register operation specified in the instructions in each of the plurality of threads; and

wherein the processing slice executes the instructions from more than one of the plurality of threads concurrently in a clock cycle.

28. The processing system of claim 27 wherein the peripheral unit is one of an input device and an output device.

29. The processing system of claim 27 wherein the peripheral operation is one of an input operation and an output operation.

30. The processing system of claim 27 wherein the command message includes at least one of a message content, a peripheral address identifying the peripheral unit, and a command code specifying the peripheral operation.

31. The processing system of claim 27 wherein the peripheral information includes a response message sent from the peripheral unit to the processing slice, the response message indicating the peripheral operation is completed.

32. The processing system of claim 31 wherein the response message includes at least one of a thread identifier identifying the first thread, an operation result of the peripheral operation, a data register address specifying a data register in the processing slice to store the operation result, and a length indicator indicating length of the response message.

33. The processing system of claim 32 wherein the peripheral bus comprises:
a bi-directional bus to transfer the command message from the processing slice to the peripheral unit and the response message from peripheral unit to the processing slice.

34. The processing system of claim 27 wherein the processing slice disables the first thread after sending the command message if the command message is a wait instruction.

35. The processing system of claim 27 wherein the first thread continues to execute after sending the command message if the command message is a non-wait instruction.

36. The processing system of claim 34 wherein the processing slice enables the first thread after receiving the response message from the peripheral unit if the first thread was disabled.

37. The processing system of claim 27 wherein the processing slice comprises:

an instruction processing unit to process instructions fetched from a program memory; and

a thread control unit coupled to the instruction processing unit to manage initiating and termination of at least one of the plurality of threads.

38. The processing system of claim 37 wherein the processing slice further comprises:

a memory access unit coupled to the instruction processing unit to provide access to one of the plurality of data memories via the data memory switch, the memory access unit having a plurality of data base registers, each of the data base registers corresponding to each of the threads; and

a register file coupled to the instruction processing unit and a peripheral message unit having a plurality of data registers, each of the data registers corresponding to each of the threads.

39. The processing system of claim 38 wherein the instruction processing unit comprises:

an instruction fetch unit to fetch the instructions from the program memory using a plurality of program counters, each program counter corresponding to each of the threads;

an instruction buffer coupled to the instruction fetch unit to hold the fetched instructions; and

an instruction decoder and dispatcher coupled to the instruction buffer to decode the instructions and dispatch the decoded instructions to one of the memory access unit, the functional unit, and the peripheral unit.

40. A processing system comprising:

a plurality of multi-thread processors;

a plurality of peripheral units;

a peripheral bus coupled to the peripheral units to transfer peripheral information between the multi-thread processors and the peripheral units, the peripheral information including a command message sent from one of the multi-thread processors to one of the peripheral units;

wherein each processor comprises a plurality of processing slices to execute a plurality of threads comprising instructions including the command message;

wherein each processing slice comprises a functional unit to perform a register operation specified in the instructions in each of the plurality of threads; and

wherein the processing slice is capable of executing the instructions from more than one of the plurality of threads concurrently in a clock cycle.

41. A processing system comprising:

a multi-thread processor having program base registers and data base registers;

at least one peripheral unit;

a peripheral bus coupled to the at least one peripheral unit to transfer peripheral information between the multi-thread processor and the at least one peripheral unit, the peripheral information including a command message sent from the multi-thread processor to the peripheral unit;

wherein the processor comprises a plurality of processing slices to execute a plurality of threads comprising instructions including the command message;

wherein each processing slice comprises a functional unit to perform a register operation specified in the instructions in each of the plurality of threads; and

wherein the processing slice is capable of executing the instructions from more than one of the plurality of threads concurrently in a clock cycle.